Finite Labs **Essentials**

Overview

> DISCLAIMER: This software is neither affiliated with nor endorsed by Control4.

Control4 programming offers only basic operations on variables: set a value, randomize, increment, decrement, or copy from another variable. Variable Expressions extends this with named expressions: build a string from one or more variables, or evaluate a mathematical equation that references one or more variables. Each expression publishes its own result variable and event for use elsewhere in programming, and can recalculate automatically when the variables it references change.

Index

- System Requirements
- Features
- Installer Setup
  - DriverCentral Cloud Setup
  - Adding the Driver
  - Expressions Tab
  - Driver Properties
    - Cloud Settings
    - Driver Settings
  - Driver Actions
- Programming
  - Referencing Variables
  - Commands
  - Output Variables and Events
  - Math Functions
  - Examples
- Developer Information
- Support
- Changelog

System Requirements

- Control4 OS 3.3+

Features

- Named expressions managed from a dedicated Expressions tab in Composer, with a searchable variable browser, insert at cursor, syntax highlighting, and a live result preview
- Each expression publishes its own `<Name> Result` variable and fires a `<Name> Calculated` event, so multiple calculations never collide
- Automatic recompute: the driver watches the variables an expression references and recalculates when they change, debounced so chatty sources (power meters) coalesce into one recalculation per second
- Equation mode evaluates math; String mode builds text from variables
- Reference any variable in the project by device id and variable id or name
- Full Lua math library plus common helpers (`abs`, `min`, `max`, `round`, ...) and conditionals (`and`, `or`, comparisons)
- Rendered expression shows each reference as `[Room > Device > Variable]`
- Expressions and last results persist across driver restarts

Installer Setup

DriverCentral Cloud Setup

> If you already have the DriverCentral Cloud driver installed in your project you can continue to Adding the Driver.

This driver relies on the DriverCentral Cloud driver to manage licensing and automatic updates. If you are new to using DriverCentral you can refer to their Cloud Driver documentation for setting it up.

Adding the Driver

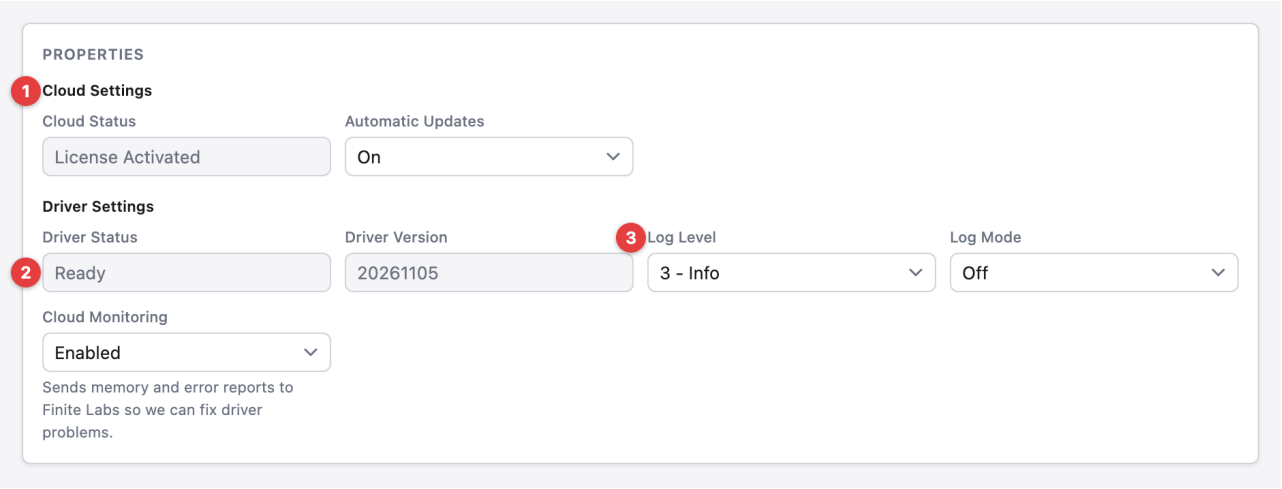
1. Download the latest `control4-finite-labs-essentials.zip` from DriverCentral.
2. Extract and install the `variable_expressions.c4z` driver.

3. Use the "Search" tab to find "Variable Expressions" and add it to your project.

Expressions Tab

The Expressions tab in Composer (select the driver, then the Expressions tab) is the primary way to create and manage standing calculations.

Properties

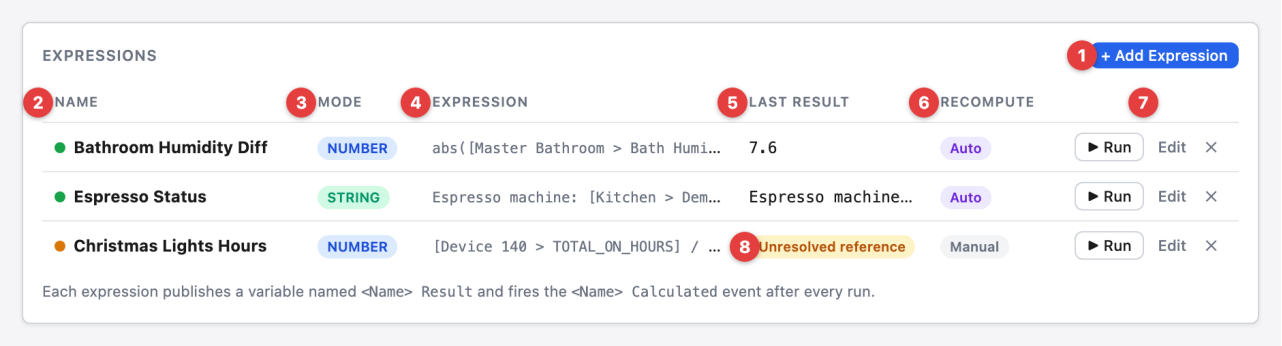


1. **Cloud Settings** - the driver's properties under the same headings as Composer's "Properties" tab.
2. **Driver Status** - grayed out, as every read-only property is: the driver sets it.
3. **Log Level** - a change applies at once, and shows in Composer's "Properties" tab too. A change made there shows here.

The Support Code is entered in Composer's "Properties" tab only.

Expression List

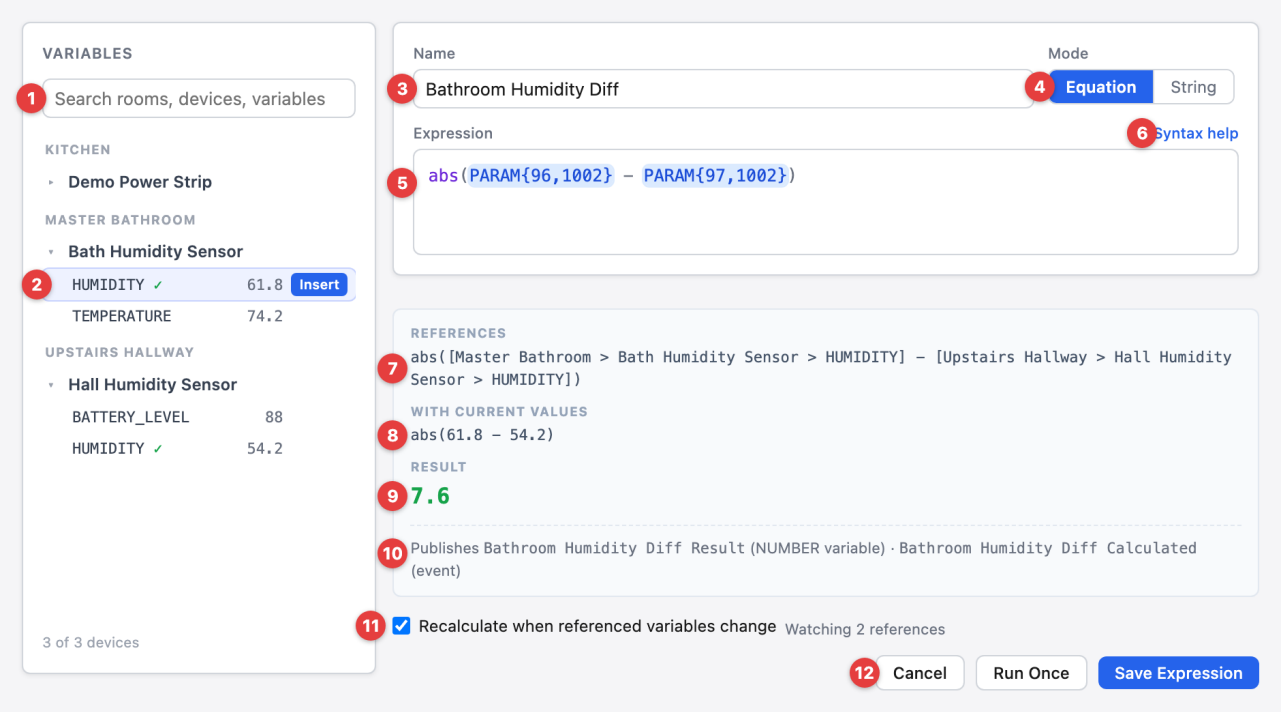
Every named expression appears as a row with its mode, a rendered view of what it references, the last result, and whether it recalculates automatically. Results update live as the driver recalculates.



1. **+ Add Expression** - opens an empty editor for a new expression.
2. **Name** - the name the output variable, the event, and programming use. The dot to its left is green once the expression has produced a result, amber when the last run failed, and grey before its first run.
3. **Mode** - `NUMBER` for an equation, `STRING` for a string template.
4. **Expression** - the template with each reference rendered as `[Room > Device > Variable]`, so a row is readable without opening it.
5. **Last Result** - the value currently published to the `<Name> Result` variable. Hovering shows how long ago it was calculated.
6. **Recompute** - `Auto` when the driver watches the referenced variables, `Manual` when the expression only runs on demand.
7. **Run / Edit / X** - recalculate now, open the expression in the editor, or delete it along with its output variable and event.
8. **Unresolved reference** - the expression references something that no longer exists, for example a device that was removed. The previous result is kept.

Expression Editor

Add Expression and Edit both open the editor in place of the list.



1. **Search** - filters the variable browser by room, device, agent, or variable name. Matching devices expand automatically.
2. **Variable** - click a device or agent to load its variables with their current values, then click a variable to insert its `PARAM{}` token at the cursor. A green check marks variables the expression already references.
3. **Name** - names the `<Name> Result` variable and the `<Name> Calculated` event. Renaming an expression renames both.
4. **Mode** - `Equation` evaluates the template as math, reading each token as its variable's current value, and publishes a NUMBER. `String` publishes the text with each token's value in place as a STRING.
5. **Expression** - the template. Tokens, functions, numbers, and strings are highlighted as you type.
6. **Syntax help** - the reference for tokens, operators, and functions, without leaving the editor.
7. **References** - the template with each token rendered as `[Room > Device > Variable]`, which is the quickest way to confirm a token points where you meant it to.
8. **With current values** - the same template with the variables' current values in place, to show what it reads (Equation mode only). A value is read as data, so this is a view of the values and not the text that runs.
9. **Result** - the value the controller computed with those values. Errors appear here in red, before anything is saved.
10. **Publish line** - the variable and event this expression owns once saved.

11. **Recalculate when referenced variables change** - when enabled, the driver watches every variable the expression references and recalculates on change, debounced by 1 second so rapidly updating sources coalesce into a single recalculation, at least every 11 seconds or so while they keep changing. The count of watched references is shown alongside. When disabled, the expression only recalculates from the Calculate Expression programming command or the Run button.
12. **Cancel / Run Once / Save Expression** - discard the changes, evaluate the expression without saving it, or save it.

Outputs

Saving an expression named `Bathroom Humidity Diff` creates:

- A read-only variable `Bathroom Humidity Diff Result`
- An event `Bathroom Humidity Diff Calculated` that fires after every successful recalculation

Renaming an expression renames its outputs; deleting it removes them.

Driver Properties

Cloud Settings

Cloud Status (read-only)

Displays the DriverCentral cloud license status.

Automatic Updates [Off | *On*]

Enables or disables automatic driver updates via DriverCentral. Changing it on any item of this driver changes it on all of them.

Driver Settings

Driver Status (read-only)

Displays the current status of the driver.

Driver Version (read-only)

Displays the current version of the driver.

Log Level [0 - Fatal | 1 - Error | 2 - Warning | *3 - Info* | 4 - Debug]

Sets the logging level. Default is `3 - Info`. While a Support Code has Composer's Lua tab open, it also offers `5 - Trace` and `6 - Ultra`.

Log Mode [*Off* | Print | Log | Print and Log]

Sets the logging mode. Default is `Off`.

Support Code

Leave it blank unless Finite Labs support gives you a code: for an hour, it opens Composer's Lua tab to this driver and adds `5 - Trace` and `6 - Ultra` to Log Level.

Cloud Monitoring [Disabled | *Enabled*]

When Enabled, the driver sends Finite Labs a short report once an hour, and a report of each error it hits, at most 10 a minute. We use them to find and fix problems in the driver. Set it to Disabled to send nothing.

A report holds the controller's MAC address and OS version, the driver's name, version, device ID and DriverCentral license status, how much memory the driver uses, and the sizes of its largest internal tables. An error report also holds the error message and where in the driver it happened. Before a message is sent, anything in it that looks like a password, token, key, network address or email address is removed.

Driver Actions

Reset Driver

Deletes every expression along with its output variable and event, and clears all persisted state.

It also turns Cloud Monitoring back on, as a fresh install has it.

Parameters:

- **Are You Sure?** [***No*** | Yes] - Confirmation to reset the driver.

Diagnostics Snapshot

Prints a report in the Lua output, and sends it to Finite Labs only while Cloud Monitoring is Enabled.

Programming

Referencing Variables

Reference a variable anywhere in a string or equation with a `PARAM{}` token:

```
PARAM{DEVICE_ID,VARIABLE_ID}
```

`DEVICE_ID` is the numeric id of the device that owns the variable. `VARIABLE_ID` is either the numeric id of the variable or its name (matched exactly, case-sensitive), for example `PARAM{96,1002}` or `PARAM{96,Humidity}`. Any variable type may be referenced, including strings, numbers, and booleans.

In an equation a token reads its variable as a value: a number as a number, `true` and `false` as booleans, and anything else as text, so `PARAM{96,Mode} == "On"` compares the text. A value is only ever data: it can't break an expression or run as part of one.

You rarely need to build a token by hand: click a variable in the Expressions tab browser and its token is inserted at the cursor. If a referenced variable cannot be found, the expression's row shows `Unresolved reference` and the previous result is kept.

Commands

Calculate Expression

Recalculate a named expression now. The Expression dropdown lists every expression defined in the Expressions tab. Useful for manual-recompute expressions or to force a recalculation at a known point in programming.

Expressions are evaluated in a sandbox that exposes only the math functions listed below, so an expression cannot read or change the rest of the driver.

Output Variables and Events

Each expression owns a read-only `<Name> Result` variable and a `<Name> Calculated` event. Recalculation is asynchronous, so trigger follow-up programming from the expression's variable-changed event (`When <Name> Result changes`) or from the `<Name>`

`Calculated` event. The event fires after every successful recalculation, even when the computed value is unchanged, which makes it useful when a variable-changed event would not re-fire.

Math Functions

Equations may use the full Lua `math` and `string` libraries (for example `math.floor`, `math.random`, `string.format`). The following helpers are also available without a prefix:

`abs`, `ceil`, `floor`, `sqrt`, `min`, `max`, `round(x, digits)`, `c2f`, `f2c`, `pi`, `huge`, `tonumber`, `tostring`

Operators follow Lua syntax: `+`, `-`, `*`, `/`, `%` (modulo), `^` (power), and parentheses for grouping. Comparisons (`==`, `~=`, `<`, `<=`, `>`, `>=`) combined with `and`, `or`, and `not` allow conditionals:

```
PARAM{96,1002} > 50 and 1 or 0
```

The same reference is also available from the Syntax help link in the expression editor.

Examples

Difference between two humidity sensors (device 96 and 97, variable 1002):

1. In the Expressions tab, add an expression named `Bathroom Humidity Diff` in Equation mode with automatic recompute enabled: `abs(PARAM{96,1002} - PARAM{97,1002})`
2. The driver recalculates it whenever either humidity changes.
3. Program off `When Bathroom Humidity Diff Result changes` (for example, run a fan when the difference exceeds a threshold).

Average of three temperatures:

```
round((PARAM{96,1001} + PARAM{97,1001} + PARAM{98,1001}) / 3, 1)
```

Status string for a custom label (String mode):

```
Pool PARAM{120,2001}F / Spa PARAM{120,2002}F
```

Developer Information



Copyright © 2026 Finite Labs LLC

All information contained herein is, and remains the property of Finite Labs LLC and its suppliers, if any. The intellectual and technical concepts contained herein are proprietary to Finite Labs LLC and its suppliers and may be covered by U.S. and Foreign Patents, patents in process, and are protected by trade secret or copyright law. Dissemination of this information or reproduction of this material is strictly forbidden unless prior written permission is obtained from Finite Labs LLC. For the latest information, please visit https://drivercentral.io/platforms/control4-drivers/utility/finite-labs-essentials

Support

If you have any questions or issues integrating these drivers with Control4, you can contact us at driver-support@finitelabs.com or call/text us at +1 (949) 371-5805.

Changelog

v20261112 - 2026-11-12

Added

- Added the Device Programmer, Light Relay, Network Requests, Sensor Aggregator, Sensor Multiplexer and Variable Expressions drivers to DriverCentral.